UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/725,016	12/02/2003	Serguei M. Beloussov	2230.0020000/MBR/GSB	3183

54089 7590 01/22/2007
BARDMESSER LAW GROUP, P.C.
910 17TH STREET, N.W.
SUITE 800
WASHINGTON, DC 20006

EXAMINER	
NGUYEN, PHILLIP H	
ART UNIT	**PAPER NUMBER**
2191	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	01/22/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/725,016	BELOUSSOV ET AL.	
	Examiner	Art Unit	
	Phillip H. Nguyen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 December 2003*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-57,59-64 and 66-73* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-57,59-64 and 66-73* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on November 09, 2006.

2. Per Applicant's request, claims 1-3, 26, 27, 40, 53-57, 59, 60-64 and 66-71 are amended.

3. Per Applicant's request, claims 58 and 65 are canceled and claims 72 and 73 are newly added.

4. Claims 1-57, 59-64, and 66-73 are pending.

Examiner Note:

Applicant appears to invoke 35 U.S.C. 112 6th paragraph in claims 53, 61-63, 65, 67, 68 by using "means-plus-function" language. However, Examiner notes that the claims recite sufficient structure, which is "computer program code" for performing those recited functions. While the claims pass the first of the three-prong test used to determine invocation of paragraph 6th, since they also recite sufficient structure within the claims to perform entirely recited functions, the claims are not in means-plus-function format, even if the claims use the term "means". Therefore, 35 U.S.C. 112 6th paragraph has not been invoked when considered these claims below.

Examiner accidentally inserted the paragraph under 35 U.S.C. 112 rejections on previous action. Examiner meant to inform the Applicant that he does not invoke 112 6th Paragraph.

Claim Objections

5. The amendment filed on November 09, 2006 overcomes the objection of claims 54, 59, 60 and 64 of the previous action. Therefore, the objection is withdrawn.

Claim Rejections - 35 USC § 112

6. The amendment filed on November 09, 2006 overcomes the rejection to claims 2, 3, and 26 of the previous action. Therefore, the rejection is withdrawn.

Response to Amendment

7. The amendment filed November 9, 2006 is objected to under 35 U.S.C. 132(a) because it introduces new matter into the claims. 35 U.S.C. 132(a) states that no amendment shall introduce new matter into the disclosure of the invention. The added material which is not supported by the original disclosure is as follows: The amended claims 1, 27, 40, 53, 67-71 recite "**the original instructions are part of the instruction set of the processor available to a user**", which is new matter. There is no support or written description in the specification for this newly added material.

Applicant is required to cancel the new matter in the reply to this Office Action.

Response to Arguments

8. Applicant's arguments filed November 09, 2006 have been fully considered but they are not deemed persuasive.

Applicant asserts on pages 17-18 of the amendment that Mahalingaiah does not disclose "patching" of the instructions at run-time as they are being executed, instead directed to patching of microcode instructions.

Examiner respectfully disagrees with all of the allegations as argued. Examiner, in his previous office action, pointed out locations in the cited prior art that matched the claimed limitations. It is reasonable to interpret the "microcode instruction" is one type of instruction code by one of ordinary skill in the art. Further more, Mahalingaiah's approach takes place at runtime as disclosed in Col 5, line 20-29 "prefetch/predecode unit 12 is coupled to receive instructions from a main memory subsystem (not shown), and is further coupled to instruction cache 6 and branch prediction unit 14...General speaking, an instruction storage device is a memory device configured to store instruction to be executed by a microprocessor". This means, while the processor is running (e.g., run-time), it receives instructions from the storage device and detects instruction to be patched and so forth.

Examiner is entitled to give claim limitations their broadest reasonable interpretation in light of the specification. See MPEP 2111 [R-1] Interpretation of Claims-Broadest Reasonable Interpretation. During patent examination, the pending claims must be 'given the broadest reasonable interpretation consistent with the specification.' Applicant always has the opportunity to amend the claims during the prosecution and broad interpretation by the examiner reduces the possibility that the

claim, once issued, will be interpreted more broadly than is justified. In re Prater, 162 USPQ 541, 550-51 (CCPA 1969).

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1-3, 6, 7, 13-17, 19-29, 31, 32, 35-45, 48, 49, 51-55, 57, 61-64, and 66-73 are rejected under 35 U.S.C. 102(b) as being anticipated by Mahalingaiah et al. (United States Patent No.: 5,983,337).

As per claim 1:

Mahalingaiah discloses a method of on-the-fly patching of executable code comprising:

- identifying original instructions to be changed ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48) while the original instructions are being executed on a processor (**in order for the prefetch/predecode unit 12 to detect instructions to be patched, the microprocessor must be in run-time environment**);

- copying the original instructions to a storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59)
- adding a jump instruction to the copied instructions to return to a next instruction after the original instruction ("**a branch instruction may be executed having a target address which lies within a cache line.**" Col 6, line 10-12); and
- replacing the original instructions while the original instructions are in the process of being executed on the processor with mark instructions ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

As per claim 2:

Mahalingaiah discloses the method as in claim 1 above, but does ~~not explicitly~~ disclose:

- prior to the copying step, allowing a write operation on a page in memory wherein the original code is located. **It is inherent in Mahalingaiah's approach in order for the prefetch/predecode unit 12 to copy the instruction to the instruction cache 16.**

As per claim 3:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- prior to the allocating step, masking interrupts ("**the instruction identification information for each instruction includes. (i) start and end pointers identifying... (ii) a valid mask containing eight bits, one for each of the eight bytes**" Col 23, line 21-25).

As per claim 6:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- wherein the original instructions are changed in reverse order ("**when packing a MROM instruction and a fast path instruction, the first of the instruction in program order is mark as the last instruction.**" Col 26, line 23-25).

As per claim 7:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- wherein the mark instructions are the same length, in bytes, as the instructions to be changed ("**each instruction block comprises a predefined number of instruction bytes... the predefined number of instruction bytes comprises eight instruction bytes stored in contiguous main memory storage location**" Col 6, line 18-34).

As per claim 9:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- wherein the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed ("**instruction cache 16 provides an indication of the instruction address being fetched**" Col 8, line 64-66, **this means, the instructions are tagged to indicate they have been fetch to cache and executed by the microprocessor**).

As per claim 13:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- enabling functionality of the copied instructions at the storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59).

As per claim 14:

Mahalingaiah discloses the method as in claim 13 above; and further discloses:

- the enabling step comprises reconciling addressing in the instructions in the storage location ("**the positions store instructions identification information and are maintained such that the instruction identification information for the first valid instruction within the subqueue is stored in a first position within the subqueue, instructions identification**

information regarding the second valid instruction (in program order) is stored in a second position within the subqueue, etc." Col 6, line 52-58).

As per claim 15:

Mahalingaiah discloses the method as in claim 14 above; and further discloses:

- wherein the enabling step comprises alignment of instructions in the instructions at the storage location ("**align instructions from instruction cache 16**" Col 6, line 5-6).

As per claim 16:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- verifying that the original code is susceptible ("**valid**") to patching ("**if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 52-53).

As per claim 17:

Mahalingaiah discloses the method as in claim 16 above; and further discloses:

- wherein the verifying step determines whether any mark instructions are already present in the original instructions ("**if an instruction opcode matches a valid opcode in the patch opcode register, prefetch/predecode unit 12 marks the instruction as a MROM**

instruction" Col 5, line 52-53, **this means, if they are matched, the mark instructions are already present in the original instructions**).

As per claim 19:

Mahalingaiah discloses the method as in claim 16 above; and further discloses:

- wherein the verifying step determines whether the original instructions include a suitable jump point that can be modified to the transfer of control to the hook ("**instruction alignment unit 18 identifies the instructions to be executed. Instruction alignment unit 18 conveys the instructions, in predicted program order**" Col 6, line 5-20).

As per claim 20:

Mahalingaiah discloses the method as in claim 16 above; and further discloses:

- wherein the verifying step determines whether the original instructions represent valid instructions ("**if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 52-53).

As per claim 21:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- placing the hook in the memory ("**decode units 20 for decode and execution**" Col 6, line 17-18).

As per claim 22:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- the hook has been previously placed in memory (**the decode units 20 is part of the microprocessor and therefore, it has been previously stored in the memory**).

As per claim 23:

Mahalingaiah discloses the method of as in claim 1 above; and further discloses:

- wherein the replacing step use an atomic write to replace the original instructions ("**an instruction scanning unit within instruction cache 16 separates the instructions fetched into instruction blocks**" Col 6, line 20-22).

As per claim 24:

Mahalingaiah discloses the method as in claim 23 above; and further discloses:

- wherein the atomic write replaces one instruction at a time ("**comparing an instruction opcode to opcode stored in an opcode patch register**" Col 5, line 48-49, **this means, comparing one instruction at a time and replacing one instruction at a time**).

As per claim 25:

Mahalingaiah discloses the method as in claim 23 above; and further discloses:

- wherein the atomic write replaces multiple instructions at a time (**an instruction scanning unit within instruction cache 16 separates the instructions fetched into instruction blocks** Col 6, line 20-22, **this means, replacing a block of instruction at a time**).

As per claim 26:

mahalingaiah discloses the method as in claim 23 above; and further discloses:

- wherein, for Intel X32 architecture, the atomic write uses any of "xchg," "lock cmpxchg8b," "lock cmpxchg," and "lock xchg" instructions ("**CMPXCHG", "CMPXCHG8B", "XCHG**" Col 28; Table 1).

As per claim 27:

Mahalingaiah discloses a method of on-the-fly patching of executable code comprising:

- verifying that original instructions to be changed are susceptible to patch ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48) while the original instructions are being executed on a processor (**in order for the prefetch/predecode unit 12 to detect instructions to be patched, the microprocessor must be in run-time environment**);
- generating pseudooriginal code from the original instructions at different storage location from the original instructions ("**instructions are fetched**

from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12" Col 7 line 58-59);

- adding a jump (branch) instruction to the pseudooriginal code to return to a next instruction after the original instructions ("**a branch instruction may be executed having a target address which lies within a cache line**." Col 6, line 10-12); and
- replacing the original code while the original code is in the process of being executed on the processor with tag instructions that indicate only their execution ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

As per claim 28:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- wherein the original instructions are changed in reverse order ("**when packing a MROM instruction and a fast path instruction, the first of the instruction in program order is mark as the last instruction**." Col 26, line 23-25).

As per claim 29:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- wherein the tag instructions are the same length, in bytes, as the instructions to be changed ("**each instruction block comprises a predefined number of instruction bytes... the predefined number of instruction bytes comprises eight instruction bytes stored in contiguous main memory storage location**" Col 6, line 18-34).

As per claim 31:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed ("**instruction cache 16 provides an indication of the instruction address being fetched**" Col 8, line 64-66, **this means, the instructions are tagged to indicate they have been fetch to cache and executed by the microprocessor**).

As per claim 32:

Mahalingaiah discloses the method as in claim 31 above; and further discloses:

- wherein the resolver determines a number of instructions that had already been executed using mark instructions ("**instruction cache 16 provides an indication of the instruction address being fetched**" Col 8, line 64-66, **this means, the instructions are tagged to indicate they have been fetch to cache and executed by the microprocessor**).

As per claim 35:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- reconciling addressing in the instructions in the storage location ("**the positions store instructions identification information and are maintained such that the instruction identification information for the first valid instruction within the subqueue is stored in a first position within the subqueue, instructions identification information regarding the second valid instruction (in program order) is stored in a second position within the subqueue, etc**." Col 6, line 52-58).

As per claim 36:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- verifying that the original code is susceptible to patching ("**if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 52-53).

As per claim 37:

Mahalingaiah discloses the method as in claim 36 above; and further discloses:

- verifying step determines whether any tag instructions are already present in the original instructions ("**if an instruction opcode matches a valid opcode in the patch opcode register, prefetch/predecode unit 12 marks the instruction as a MROM instruction**" Col 5, line 52-53, **this means, if they**

are matched, the mark instructions are already present in the original instructions).

As per claim 38:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- placing the hook in the memory ("**decode units 20 for decode and execution**" Col 6, line 17-18).

As per claim 39:

Mahalingaiah discloses the method as in claim 27 above; and further discloses:

- wherein the replacing step use an atomic write to replace the original instructions instructions ("**an instruction scanning unit within instruction cache 16 separates the instructions fetched into instruction blocks**" Col 6, line 20-22).

As per claim 40:

Mahalingaiah discloses a method of on-the-fly patching of executable code comprising:

- identifying original instructions to be changed ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48) while the original instructions are being executed on a processor (**in order for the**

prefetch/predecode unit 12 to detect instructions to be patched, the microprocessor must be in run-time environment);

- allocating a storage location for storing a functionally equivalent copy of the original instructions ("**instruction cache 16 is a high speed cache memory provided to store instructions**" Col 7, line 39-40);
- copying the original instructions to a storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);
- replacing the original instructions while the original instructions are in the process of being executed on the processor with mark instructions ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

As per claim 41:

Mahalingaiah discloses the method as in claim 40 above, but doe not explicitly disclose:

- prior to the allocating step, allowing a write operation on a page in memory where the original instructions are located. **It is inherent in Mahalingaiah's approach in order for the prefetch/predecode unit 12 to copy the instruction to the instruction cache 16.**

As per claim 42:

Mahalingaiah discloses the method as in claim 40 above; and further discloses:

- adding a jump instruction to the copied instructions to return to a next instruction after the original instructions ("**the instruction identification information for each instruction includes. (i) start and end pointers identifying... (ii) a valid mask containing eight bits, one for each of the eight bytes**" Col 23, line 21-25).

As per claim 43:

Mahalingaiah discloses the method as in claim 40 above; and further discloses:

- Wherein the original instructions are changed in reverse order ("**when packing a MROM instruction and a fast path instruction, the first of the instruction in program order is mark as the last instruction.**" Col 26, line 23-25).

As per claim 44:

Mahalingaiah discloses the method as in claim 1 above; and further discloses:

- wherein the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed ("**instruction cache 16 provides an indication of the instruction address being fetched**" Col 8, line 64-66, **this means, the instructions are**

tagged to indicate they have been fetch to cache and executed by the microprocessor).

As per claim 45:

Mahalingaiah discloses the method as in claim 44 above; and further discloses:

- wherein the resolver (**scanning unit within the instruction cache 16**) determines a number of instructions that had already been executed using mark instructions ("**instruction cache 16 provides an indication of the instruction address being fetched**" Col 8, line 64-66**, this means, the instructions are tagged to indicate they have been fetch to cache and executed by the microprocessor**).

As per claim 48:

Mahalingaiah discloses the method as in claim 45 above; and further discloses:

- verifying that the original code is susceptible to patching ("**if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 52-53).

As per claim 49:

Mahalingaiah discloses the method as in claim 48 above; and further discloses:

- wherein the verifying step determines whether any mark instructions are already present in the original instructions ("**if an instruction opcode**

matches a valid opcode in the patch opcode register, prefetch/predecode unit 12 marks the instruction as a MROM instruction" Col 5, line 52-53, **this means, if they are matched, the mark instructions are already present in the original instructions**).

As per claim 51:

Mahalingaiah discloses the method as in claim 40 above; and further discloses:

- wherein the replacing step use an atomic write to replace the original instructions ("**an instruction scanning unit within instruction cache 16 separates the instructions fetched into instruction blocks**" Col 6, line 20-22).

As per claim 52:

Mahalingaiah discloses the method as in claim 40 above; and further discloses:

- enabling functionality of the copied instructions at the storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59).

As per claim 53:

Mahalingaiah discloses a computer useable medium having computer program logic stored thereon for executing on a processor, for on-the-fly patching of executable code, the computer logic comprising:

- computer program code means for identifying original instructions to be changed while the original instructions are being executed on a processor changed ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48);
- computer program code means for copying the original instructions to a storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);
- computer program code means for adding a jump instruction to the copied instructions to return to a next instruction after the original instructions ("**a branch instruction may be executed having a target address which lies within a cache line**." Col 6, line 10-12); and
- computer program code means for replacing the original instructions while the original instructions are in the process of being executed on the processor with mark instructions ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

Note: The word "for" in the preamble and the body of the claim indicates intended use, which does not carry patentable weight. The limitations following the

phrase "for" describe only the intended use but not necessarily required functionality of the claim.

As per claim 54:

Mahalingaiah discloses the computer program logic as in claim 53 above; and further discloses:

- wherein the original instructions are changed in reverse order ("**when packing a MROM instruction and a fast path instruction, the first of the instruction in program order is mark as the last instruction.**" Col 26, line 23-25).

As per claim 55:

Mahalingaiah discloses the computer program logic as in claim 53 above; and further discloses:

- wherein the mark instructions are the same length, in bytes, as the instructions to be changed ("**each instruction block comprises a predefined number of instruction bytes... the predefined number of instruction bytes comprises eight instruction bytes stored in contiguous main memory storage location**" Col 6, line 18-34).

As per claim 57:

Mahalingaiah discloses the computer program logic as in claim 53 above; and further discloses:

- wherein the modified instructions include a resolver to determine a number of the instructions at a location of the original code that had already been executed ("**instruction cache 16 provides an indication of the instruction address being fetched**" Col 8, line 64-66, **this means, the instructions are tagged to indicate they have been fetch to cache and executed by the microprocessor**).

As per claim 61:

Mahalingaiah discloses the computer program logic as in claim 53 above; and further discloses:

- computer program code means for enabling functionality of the copied instructions at the storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59).

As per claim 62:

Mahalingaiah discloses the computer program logic as in claim 61 above; and further discloses:

- wherein the computer product code means for enabling functionality of the copied instructions at the storage location comprises computer product code means for reconciling addressing in the instructions in the storage location ("**the positions store instructions identification information and are maintained such that the instruction identification information for the first valid instruction within the subqueue is stored in a first position within the subqueue, instructions identification information regarding the second valid instruction (in program order) is stored in a second position within the subqueue, etc**." Col 6, line 52-58).

As per claim 63:

Mahalingaiah discloses the computer program logic as in claim 53 above; and further discloses:

- computer program code means for verifying that the original code is susceptible to patching ("**if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 52-53).

As per claim 64:

Mahalingaiah discloses the computer program logic as in claim 63 above; and further discloses:

- wherein the computer program code means for verifying determines whether any mark instructions are already present in the original instructions ("**if an instruction opcode matches a valid opcode in the patch opcode**

register, prefetch/predecode unit 12 marks the instruction as a MROM instruction" Col 5, line 52-53, **this means, if they are matched, the mark instructions are already present in the original instructions**).

As per claim 66:

Mahalingaiah discloses the computer program logic as in claim 53 above; and further discloses:

- wherein the computer program code means for replacing step use an atomic write to replace the original instructions ("**an instruction scanning unit within instruction cache 16 separates the instructions fetched into instruction blocks**" Col 6, line 20-22).

As per claim 67:

Mahalingaiah discloses a computer useable medium having computer program logic stored thereon for executing on a processor, for on-the-fly patching of executable code, the computer program logic comprising:

- computer program code means for verifying that original instructions to be changed are susceptible to patching while the original instructions are being executed on a processor ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48);
- computer program code means for generating pseudooriginal code from the original instructions at a different storage location from the original

instructions ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);

- computer program code means for adding a jump (branch) instruction to the pseudooriginal code to return to a next instruction after the original instructions ("**a branch instruction may be executed having a target address which lies within a cache line**." Col 6, line 10-12); and
- computer program code means for replacing the original code while the original code is in the process of being executed on the processor with tag instructions that indicate only their execution ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

Note: The word "for" in the preamble and the body of the claim indicates intended use, which does not carry patentable weight. The limitations following the phrase "for" describe only the intended use but not necessarily required functionality of the claim.

As per claim 68:

Mahalingaiah discloses a computer useable medium having computer program logic stored thereon for executing on a processor, for on-the-fly patching of execution code, the computer program logic comprising:

- computer program code means for identifying original instructions to be changed are susceptible to patch ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48), while the original instructions are being executed on a processor;
- computer program code means for allocating a storage location for storing a functionality copy of the original instructions ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);
- computer program code means for copying the original instructions to the storage location ("**a branch instruction may be executed having a target address which lies within a cache line**." Col 6, line 10-12); and
- computer program code means for replacing the original instructions with mark instructions ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

Note: The word "for" in the preamble and the body of the claim indicates intended use, which does not carry patentable weight. The limitations following the

phrase "for" describe only the intended use but not necessarily required functionality of the claim.

As per claim 69:

Mahalingaiah discloses a system for on-the-fly patching of executable code comprising:

- means for identifying original instructions to be changed ("**prefetch/predecode unit 12 detects instructions to be patched**" Col 5, line 47-48) while the original instructions are executed on a processor;
- means for copying the original instructions to a storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);
- means for adding a jump instruction to the copied instructions to return to a next instruction after the original instructions ("**a branch instruction may be executed having a target address which lies within a cache line**." Col 6, line 10-12); and
- means for replacing the original code while the code is in the process of being executed on the processor with mark instructions ("**marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

Note: The word "for" in the preamble of the claim indicates intended use, which does not carry patentable weight. The limitations following the phrase "for" describe only the intended use but not necessarily required functionality of the claim.

As per claim 70:

Mahalingaiah discloses a system for on-the-fly patching of executable code comprising:

- means for verifying that original instructions to be changed are susceptible to patch while the instructions are being executed on a processor ("**prefetch/predecode unit 12 detects instructions to be patch... if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 47-53);
- means for generating pseudooriginal code from the original instructions at a different storage location from the original instructions ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);
- means for adding a jump instruction to the pseudooriginal code to return to a next instruction after the original instructions ("**a branch instruction may be executed having a target address which lies within a cache line.**" Col 6, line 10-12); and
- means for replacing the original code while the original code is in the process of being executed on the processor with tag instructions that indicate only

their execution ("**prefethc/predecode unit 12 marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

Note: The word "for" in the preamble of the claim indicates intended use, which does not carry patentable weight. The limitations following the phrase "for" describe only the intended use but not necessarily required functionality of the claim.

As per claim 71:

Mahalingaiah discloses a system for on-the-fly patching of executable code comprising:

- means for identifying original instructions to be changed while the instructions are executed on a processor ("**prefetch/predecode unit 12 detects instructions to be patch... if an instruction opcode matches a valid opcode in the patch opcode register**" Col 5, line 47-53);
- means for allocating a storage location for storing a functionally equivalent copy of the original instructions ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);
- means for copying the original instructions to a storage location ("**instructions are fetched from the main memory and stored into instruction cache 16 by prefetch/predecode unit 12**" Col 7 line 58-59);

- means for replacing the original instructions while the original instructions are in the process of being executed on the processor with mark instructions ("**prefethc/predecode unit 12 marks the instruction as an MROM instruction**" Col 5, line 53-54) and a transfer of control to a hook ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

Note: The word "for" in the preamble of the claim indicates intended use, which does not carry patentable weight. The limitations following the phrase "for" describe only the intended use but not necessarily required functionality of the claim.

As per claim 72:

Mahalingaiah discloses the method as in claim 1 above, but does not explicitly discloses:

- wherein the process of execution of the original instructions is not interrupted throughout the patching process. It is inherent in Mahalingaiah since microprocessor is still running in order for its components (prefetch/predecode unit 12, a branch prediction unit 14, instruction cache 16...) to perform the patching process.

As per claim 73:

Recites the same limitation as in claim 72 and therefore suffers the same reason set forth to claim 72.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 5, 8, 10, 11, 12, 30, 33, 34, 46, 47, 56, 59, and 60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah et al (United States Patent No.: US 5,983,337).

As per claim 5:

Mahalingaiah discloses the method as in claim 1 above, but does not explicitly disclose:

- after the replacing step, unmasking interrupts.

However, it would have been obvious to one having ordinary skill in the art at the time of the invention was made to recognize that after the replacing step, unmasking interrupts to allow the owner to see and identifying the exception errors.

Therefore, one of ordinary skill would have been motivated to include unmasking interrupts step in order to recognize the exception errors.

As per claims 8:

Mahalingaiah discloses the method as in claim 1 above, but does not explicitly disclose:

- wherein the mark instructions are shorter in length, in bytes, as the instructions to be changed, and include NOP (no operation) filler.

However, Applicant discloses that in the Intel X86 architecture in the specification, when an instruction is changed, its length is never increased (Paragraph 49). It would have been obvious to one having ordinary skill in the art at the time the invention was made to recognize that in Intel X86 architecture, any changed command is either the same length as the original instruction or is shorter with corresponding NOP instructions (no operation) in the remaining bytes.

Therefore, one would have been motivated to use this feature in the Intel X86 architecture for checking to see if the mark instructions are shorter in length, in bytes, as the instructions to be changed or any other useful reason for the invention.

As per claims 30 and 56:

Recite the same limitation as recited in claim 8 and therefore suffer the same reason set forth to claim 8.

As per claim 10:

Mahalingaiah discloses a method as in claim 9 above, but does not explicitly disclose:

- wherein the resolver determines a number of instructions that had already been executed using mark instructions.

However, Applicant discloses that in Intel X86 architecture in the specification, the resolver always know how many instructions remain to be executed by using the mark instructions (Paragraph 60).

Therefore, it would been obvious to one having ordinary skill in the art at the time the invention was made to recognized that if the number of instructions remain to be executed are always known by using the mark instructions then the number of instruction that had already been executed by using mark instructions are also known. Therefore, one would have been motivated to use the mark instructions to determine the number of mark instructions had already been executed.

As per claim 11:

Mahalingaiah discloses the method as in claim 10, but does not explicitly disclose:

- if the number of instructions that had already been executed is less than a number of original instructions to be changed, the resolver calls the copied instructions at the storage location so as to imitate a "no patch installed" scenario.

However, it would have been obvious to one having ordinary skill in the art at the time of the invention was made to recognize that after the instructions had already been executed, the contents of the instructions had already been changed. So, if there is an interrupt occurs during the execution process, the hook is going to execute before the unchanged instructions of the original code not the changed instructions. Otherwise the patching result would be different than expected.

Therefore, one would have been motivated to add this step in the patching process to ensure the patching results are correctly.

As per claims 36, 46, and 59:

Recite the same limitation as recited in claim 11 and therefore suffer the same reason set forth to claim 11.

As per claim 12:

Mahalingaiah discloses the method as in claim 11 above; and further discloses:

- wherein, after execution of the instructions at the storage location, the resolver returns control to the next instruction ("**dispatches the substitute instruction into the instruction processing pipeline**" Col 6, line 3-4).

As per claim 34, 47, and 60:

Recite the same limitation as in claim 12 above and therefore suffer the same reason set forth to claim 12.

3. Claims 4, 18, and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah et al (United States Patent No.: US 5,983,337) in view of Scott et al (United States Patent No.: US 6,615,329).

As per claim 4:

Mahalingaiah discloses the method as in claim 1 above, but does not explicitly disclose:

- after the replacing step, disallowing (disabling) a write operation on the page in memory where the block of code is located.

However, Scott discloses an analogous method that

- disable (disallow) a write operation on the page in memory where the block of code is located to protect the area from unauthorized user ("**disable write operations to the protected area**" Col 9,53-54).

Therefore, It would have been obvious to one having ordinary skill in the art at the time the invention was made to modify Mahalingaiah's approach to allow disable write operation. One of ordinary skill would have been motivated to consider protecting the memory area by disable or disallow a write operation after data have been copied from memory to storage location.

As per claims 18:

Mahalingaiah discloses the method as in claims 16 above, but does not explicitly disclose:

- determine whether any copy protect instructions are already present in the original instructions.

However, Scott discloses an analogous method having instructions within the protected area of memory ("**software instructions within the protected area of memory**" Col 5, line 18-19).

Therefore, it would have been obvious to one having an ordinary skill in the art at the time the invention was made to modify Mahalingaiah's approach to include copy protect instruction. One of ordinary skill in the art would have been motivated to recognize having instructions in the protected area to prevent the instructions from being modified by an unauthorized user.

As per claim 50:

Recites the same limitations as recite in claim 18 and therefore suffers the same reason set forth to claim 18.

Conclusion

4. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the

shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Phillip H. Nguyen whose telephone number is (571) 270-1070. The examiner can normally be reached on Monday - Thursday 10:00 AM - 3:00 PM EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

PN
01/17/2007

WEI ZHEN
SUPERVISORY PATENT EXAMINER